Local Console Alarm Display

Alarms on bottom line

R. Goodwin

Aug 9, 1989

Local Consoles can optionally (via option switch on Crate Utility Board) display alarm messages on the bottom line of the small screen display. Since there is only a single line available for this purpose—shared by the Small Memory Dump feature—there are complications which arise due to the fact that the message must be displayed long enough to be noticed by a human. This note describes the current scheme for handling such display messages.

Option switches

Three option switches relate to local alarms encoding and display.

4: Enable *analog* alarms to be included for local display.
3: Encode and output alarm message to serial port.
2: Encode and display alarm message on bottom line of screen.

For no local alarms encoding, switches #2 and #3 should be off. When switch #4 is off, only binary and comment alarms are encoded for local handling. For messages which are written to the serial port, the time-of-day associated with the alarm message is included. (The small screen does not have room to include the time-of-day, but it does indicate a “>” mark to show that the single message shown is fresh.)

Alarm message queuing

All messages which are destined to be sent to the network pass through the Output Pointer Queue, or `OUTPQ`. This includes data requests, settings, answers, and alarm messages generated by the Local Station. It can also include alarm messages generated by another station if those alarm messages are received by the Local Station. In that case, they are merely passed to the `OUTPQ` but marked (by setting the “used” bit in the queue entry) as having already been sent to the network so they aren’t sent again.

The QMonitor Task monitors all entries placed in the `OUTPQ`. It first monitors the entries to see that they have been transmitted to the network. If they take too long, they time out, in order that the queue traffic is not stalled. The time-out period is about 1 second.

After the entries have been delivered to the network, QMonitor scans the entries to check for alarm messages that should be locally displayed. If the message is an alarm message that should be displayed on the small screen, the message is encoded and passed to `ADspQEnt`. This routine could be declared as follows:

```
Function ADspQEnt(row,col: Integer;
VAR dspText: Chars32;
nChars, nCycles: Integer): Integer;
```

The row and col parameters indicate the position on the screen for the displayed

message. The dspText is the array of characters to be displayed. The nChars is the number of characters in the text. The nCycles is the delay in cycles to allow time for the message to be read before it is permitted to be overwritten by a new message. For alarm messages, the following constant values are used:

row= 15
col= 0
nChars= 32
nCycles= 30

The function value is an error status code with these possible values:

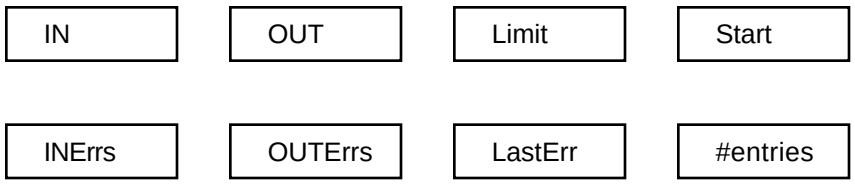
0: No errors.
1: Cannot allocate ADSPQ.
2: nChars zero or negative.
3: Cannot allocate text message block.
4: Bad ADSPQ header.
5: ADSPQ full. Cannot accept message.

The other call that QMonitor makes every cycle is to ADspQMon, which monitors the Alarm Display Queue to update the screen as needed and show the queued messages on the screen for the time designated. It has no arguments and no function return, although it can report errors through the ADSPQ header as follows:

6: Invalid OUT ptr.
7: Unexpected memory block type.

Structure of ADSPQ

The queue is created by ADspQEnt the first time it is called, and a system global variable is set to point to it. It currently has room for 124 entries. At 2 seconds per message displayed (30 cycles @ 15 Hz), it would take 4 minutes to empty a queue that was full. The queue header has the following structure:



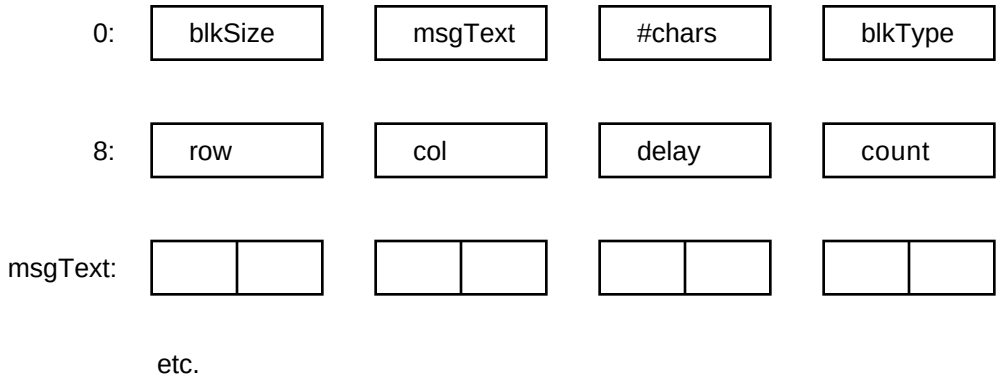
When the queue is initialized, IN and OUT and START are all set equal to the size of the header (16 bytes). LIMIT is set to the size of the queue, which is the size of the header plus the product of the number of entries (124) and the size of each entry (4 bytes). With these values, the size of the queue is 512 bytes. The other 4 words in the header are cleared.

ADspQEnt places a new entry into the queue and advances the queue's IN ptr, an offset to the next entry to be used. ADspQMon processes and removes entries from the queue as required and advances the OUT ptr. When the OUT ptr reaches LIMIT, it is

reset to the START value. When it reaches equality with IN, the queue is empty.

Diagnostics are included which count errors detected by both ADspQEnt and ADspQMon. The last error code is also recorded as is the total number of entries placed into ADSPQ.

The entry that is placed into ADSPQ is a pointer to a text message block whose format is as follows:



The first word is the allocated block size, the second is the offset to the message text, the third is the length of the text, and the fourth is the block type (=10). The next three words denote the position for the text on the screen and the delay in cycles allowed for viewing. The next word is used to count down the delay. The text to be displayed completes the contents of the block.

ADspQMon processing

Each entry placed into the ADSPQ by ADspQEnt points to a memory block that it allocated. As each block is processed by ADspQMon and displayed for the indicated delay time, the block is freed. If the option switch is turned off during the display of messages, the current message will time out its delay, and no more messages will appear. Any waiting queue entries will be skipped and the associated blocks freed.

When a new message is about to be displayed, the first two characters from the screen are read to see if the display area is available. The convention is that the first two characters should be blanks to declare this. If they are not blank, the new message is discarded and the block freed. This allows the use of the Small Memory Dump when alarm messages on the bottom line are enabled without concern that a new message will overwrite the memory data that is displayed. When the first character of the address whose memory is to be displayed is typed, no new alarms will be written there. When the feature is disabled and the line is blanked, by interrupt at the start of the line, new alarms will again be allowed to appear.

After a new message has timed out its delay, two blanks are written at the start of the display area to indicate that the message shown is no longer “fresh.” Just before the blanks are written, a check is made that the first two characters are still the same characters that were originally written there; if they are not, the blanks are not written. This again allows the user to “take over” and use the memory dump feature. Current

new messages are marked with "> " in the first two characters to indicate a "fresh" message.

Code modularization

In the modern spirit of OOP, "information hiding" principles have been followed in the implementation of this alarms display handling. The layout of `ADSPQ` and the layout of the associated message blocks are known only within the new module `ADSPQMON` which includes both routines described herein. QMonitor only knows to pass the messages it wants displayed to `ADspQEnt` and that it must call `ADspQMon` every cycle.

Serial alarms output

When QMonitor senses that a serial port version of an alarm message is to be produced, it only has to call the `PrintLn` routine with the line of text to be output. Since the serial characters are spooled to memory, there is no delay in processing the serial output.